Preface

This report contains a collection of abstracts for talks given at the "Deduktionstreffen" held at Kaiserslautern, October 6 to 8, 1993. The Deduktionstreffen is the annual meeting of the Fachgruppe Deduktionssysteme in the Gesellschaft für Informatik (GI). This Fachgruppe represents the German community of researchers in the area of automated reasoning.

The topics of the talks range from theoretical aspects of term rewriting systems and higher order resolution to descriptions of practical proof systems in various applications. They are grouped together according the following classification: Distribution and Combination of Theorem Provers, Termination, Completion, Functional Programs, Inductive Theorem Proving, Automated Theorem Proving, Proof Presentation.

It is a tradition of the Deduktiontreffen that mainly the groups at the organizing university present their work. Besides that, talks from the German speaking community are given. It is the aim of the meeting to present and discuss ongoing research. In general, the final versions of the results presented will be submitted to journals and conferences later on. So this report may give an impression what is going on for the moment in the organizing department and - to a small extent - in Germany in the field of automated reasoning.

We look forward to have an interesting meeting and fruitful discussions.

Jürgen Avenhaus

Klaus Madlener

Program

On conditional rewrite systems with extra variables and deterministic logic programs

Jürgen Avenhaus , Carlos Loría-Sáenz
Fachbereich Informatik, Universität Kaiserslautern
Postfach 3049 , 67653 Kaiserslautern
Email : {avenhaus , loria}@informatik.uni-kl.de

Abstract

Conditional rewrite systems are widely used as a high-level language to write functional programs. Often one wants to prove that such a system R is canonical, i.e. terminating and confluent. This guarantees that for any input all possible computations stop and give the same result. There are well-known methods to prove termination and confluence if no extra variables are allowed. (A variable in a rule ρ is called an extra variable if it does not appear in the left-hand side of ρ.)

Functional programming naturally demands for the *where*-construct and this construct can be incorporated into the rewrite system approach only by allowing extra variables. But extra variables should be allowed only in a very restricted form since it is not clear how to instantiate them when only the variables in the left-hand side of a rule are instantiated for rewriting. So in this paper we restrict to deterministic rewrite rules: We require that the extra variables are 'input bounded'. It is known that $\rightarrow_R$ is computable and terminating if R is quasi-reductive. We prove that $\rightarrow_R$ is confluent if R is in addition strongly deterministic and all proper critical pairs are joinable. Note that no paramodulation pairs (overlapping into the conditions) and no resolution pairs (factoring of a condition) need to be computed. These pairs may be harmful for arbitrary conditional rewrite systems with extra variables. As far as critical pairs are concerned, we neither need to consider variable overlappings nor overlappings of a rule with itself on top level. (Both are needed if R is not strongly deterministic.).

For many strongly deterministic rewrite systems R encountered in practice it can be proved that all proper critical pairs are either unfeasible or context-joinable. Then R will be confluent, provided it is quasi-reductive.

If R is a standard conditional rewrite system that is confluent then R is logical, i.e. $\stackrel{*}{\longleftrightarrow}_R$ equals the R-equality $=_R$. This is not true for a strongly deterministic R, one needs in addition the termination of R or a restriction on the right-hand sides of the condition in the rules that is more restrictive than strong determinism.

It is known that well-moded Horn clause programs can be translated into this class of rewrite systems. We show how to prove that a well-moded program is uniquely terminating, i.e., any derivation starting with a well-moded query stops and all refutations give the same answer substitution.

J. Avenhaus, C. Loría-Sáenz: Canonical conditional rewrite systems containing extra variables, SEKI-Report SR-93-03, Univ. Kaiserslautern.

Proving Termination of Rewriting Modulo a Built-in Algebra

Klaus Becker
Universität Kaiserslautern
Erwin-Schrödinger-Str. , 67663 Kaiserslautern
klbecker@informatik.uni-kl.de

Abstract

The termination problem to be considered arises with the rewrite operationalization of a specification with built-in algebra. Such a specification consists of two parts, a "base" part that introduces the predefined data types and a "top" part that partially defines new operations over the predefined data types. Formally, such a specification is given as a triple $SP = (\Sigma, \mathcal{R}, \mathcal{A})$, where $\Sigma = \Sigma_0 + \Sigma_1$ is a signature enrichment (such that Σ_1 introduces no new sorts for simplicity), $\mathcal{R}$ is a system of directed equations of type $\gamma \Rightarrow u = v$ (where γ is a Σ_0-formula and u, v are Σ-terms such that u involves at least one new function symbol) and $\mathcal{A}$ is a term-generated Σ_0-algebra. The intuition to be associated with such a specification is that $\mathcal{A}$ fixes the interpretation of the base symbols from Σ_0 and that the system $\mathcal{R}$ is to partially define the new function symbols from Σ_1 over the built-in algebra $\mathcal{A}$.

In order to model partiality we use an order-sorted algebra construction. The order-sorted algebras of interest contain an isomorphic copy of $\mathcal{A}$ as a base part. Furthermore they introduce (if necessary) auxiliary elements besides the base elements to artificially make total all new functions. The latter is achieved by introducing so-called error sorts. The auxiliary character of the non-base elements is reflected by the fact that variables are instantiated by base elements only.

In order to reason about such a specification (e.g. inductive theorem proving) we use a rewrite relation $\longrightarrow_{\mathcal{R}/\mathcal{A}}$ that is defined on Σ-ground terms as follows: Let $s \longrightarrow_{\mathcal{R}/\mathcal{A}} t$ iff there exist Σ-ground terms s', t', a rule $\gamma \Rightarrow u = v \in \mathcal{R}$ and a Σ_0-ground substitution τ such that (i) $s \sim_{\mathcal{A}} s' \equiv s'[\tau(u)] \rightarrow s'[\tau(v)] \equiv t' \sim_{\mathcal{A}} t$ and (ii) $\mathcal{A} \models \tau(\gamma)$. Here $\sim_{\mathcal{A}}$ denotes the least congruence relation on the Σ-ground terms that extends the structure induced by $\mathcal{A}$ on the Σ_0-ground terms. This rewrite relation has a well-behaved theory. For instance one easily shows that the usual syntactic critical pair lemma carries over to the semantically enriched context.

Our goal is to develop termination criteria for this rewrite relation modulo $\mathcal{A}$. For that purpose we generalize notions and ideas from usual syntactic rewriting. In particular we define a notion of reduction ordering system that provides a partial ordering $>^{(\gamma)}$ for every Σ_0-formula γ. The relation $\longrightarrow_{\mathcal{R}/\mathcal{A}}$ can be shown to be terminating whenever the compatibility condition $u >^{(\gamma)} v$ is satisfied for all $\gamma \Rightarrow u = v \in \mathcal{R}$ for a given reduction ordering system. To give an example of a reduction ordering system we generalize the well-known recursive path ordering construction such that knowledge about the built-in algebra $\mathcal{A}$ can be integrated into the construction. Details and examples can be found in:

J. Avenhaus and K. Becker, Conditional rewriting modulo a built-in algebra, SEKI Report SR-92-11.

A fine grain sort discipline and its application to program construction

Jochen Burghardt
GMD Forschungsstelle
Vincenz-Prießnitz-Straße 1
D-76131 Karlsruhe
EMail: burghard@karlsruhe.gmd.de

Abstract

A discipline of "extensional" sorts is presented allowing the introduction of a sort by recursive definition of its set of ground constructor terms in an initial algebra. During a proof, new sorts can be dynamically introduced to describe newly occurring terms. Algorithms for computing intersection, union, and difference of two sorts are given as well as for deciding the subsort and the sort equivalence property.

The signature of a non-constructor function needs not be provided externally but may be computed from its defining equations depending on the actual argument sorts, thus leading to a potentially infinite overloading and hence a corresponding precision. The signature calculation is based on a kind of "sort rewriting" similar to term rewriting, and loop checking, recognizing certain inductive invariants of a function.

Often, a non-constructor function f maps inputs starting with different constructors into disjoint range sets which are, however, too sophisticated for ordinary sort disciplines to separate them. Yet, our sort discipline does not only allow in many cases to express the range sets as disjoint sorts but is also able to compute resp. estimate them from the function equations. Thus, when solving an equation $f(x) = t$, one can determine from the sort of the term t the starting constructor of the solution for x, that is, which defining equation of f to use in a narrowing step.

It can be shown that an unsorted narrowing calculus remains complete if its rules are extended by appropriate sort restrictions. The extended calculus is able to cut off infinite branches of the search space, justifying the overhead for sort calculation.

Following the paradigm of program synthesis by equation solving this means that derivation steps contributing to the solution term may be found automatically by sort considerations. As an example, the use of the sort discipline in the implementation of sets of lists by n-ary trees with son and brother pointers is sketched.

Finally, the formalism is generalized to describe infinite sets of ground substitutions instead of ground terms, resulting in a sort discipline able to reflect variable bindings in terms, i.e. $x + x$ and $x + y$ may have different sorts.

Integration of logic functions

B. I. Dahn
Institut für Reine Mathematik der Humboldt-Universität
Ziegelstr. 13a
D-10099 Berlin
dahn@hubinf.informatik.hu-berlin.de

Abstract

Within the framework of the project "Deduction and lattice-ordered groups", provers developed within the DFG-Schwerpunkt " Deduktion" (DISCOUNT and SETHEO) and domain specific methods developed and implemented independently, have been integrated into one system ILF.

DISCOUNT (Univ. of Kaiserslautern) specializes in proving equations, SETHEO (TU Munich) represents a general resolution prover and the domain specific methods contain special proof tactics as well as the possibility of solving proof problems by testing in special structures.

All these systems work together in the framework of proof tactics. The resulting possibilities have been utilized in the construction of proofs from the theory of lattice-ordered groups. Among others there is a proof tactic that takes automatically any subproblem produced by the user and passes it to an appropriate tool.

It turned out that DISCOUNT is able to prove automatically simple equations whose proof is toilsome for man. It was not necessary to restrict the applied theories for this purpose. In order to apply SETHEO successfully, it was necessary - even for simple equations that could be handled by DISCOUNT - to reduce the theories used. However, due to its general characteristics, SETHEO could also be applied to advanced problems that use conditional equations.

In a next step, special tactics allowing the user to concentrate on essential steps in the construction of a proof were developed. For this, relatively small knowledge bases were formed and made available to the automated system working in the background. Now, in addition to the problem to be solved, the user could provide the system with a series of special further axioms, which were thus available for automatically proving the statement besides the general knowledge base. A special configuration of the knowledge base was possible, too. E. g., by restricting to monotonicity statements it could be achieved that SETHEO could be used as an efficient tool for proofs by estimation.

Recently possibilities for running proof tactics in the background have been implemented. Thus the background can be used as one large specially configured automated theorem prover that may change it's behaviour depending on the problems produced by the user.

Distributed knowledge-based theorem proving by team work

Jörg Denzinger
FB Informatik
Universität Kaiserslautern
67653 Kaiserslautern
denzinge@informatik.uni-kl.de

Abstract

Distributing the theorem proving task to several experts is a promising idea. Our approach, the team work method, allows cooperation as well as competition between several heuristics (or pieces of control knowledge) in form of so called experts. The experts work independantly for a while and then they are forced to cooperate by means of a so called team meeting. Each expert has a referee that judges the work done by the expert (competition) and that selects useful results of his expert (thus achieving cooperation) prior to the team meeting. During the team meeting the referees report to a supervisor that uses all results of the best expert and the selected results of the other experts to generate a new starting input for a next round. This is repeated until a proof is found. All components use knowledge : Experts use different tactical control knowledge, referees use assessment knowledge and the supervisor is based on strategical control knowledge.

We used the team work method to distribute equational theorem proving based on unfailing completion ([BDP89]). Experiences showed that for many examples remarkable (i.e. "super-linear") speed-ups can be achieved. We have shown completeness of this team work completion even if experts are used with heuristics that are far away from being complete themselves (see [AD93]). This allowed the development and use of goal-directed heuristics for completion that have proven to be quite sucessful in cooperation with standard strategies.

References:

[AD93] Avenhaus, J. ; Denzinger, J. :
Distributing equational theorem proving,
SEKI-Report SR-93-06, Universität Kaiserslautern, 1993.

[BDP89] Bachmair, L. ; Dershowitz, N. ; Plaisted, D.A. :
Completion without Failure,
Coll. on the Resolution of Equations in Algebraic Structures,
Austin(1987), Academic Press, 1989.

Matrices and Sequent Systems

Bertram Fronhöfer
Institut für Informatik, Technische Universität München
Arcisstr. 21
D-80290 München
fronhoef@informatik.tu-muenchen.de

Abstract

So-called 'Linear Connection Proofs'—a restriction of the Connection Method—were proposed in 1986 by W. Bibel as a logical approach to plan generation which does not suffer under the frame problem. Pretty soon an hypothesis came up, which suggested that 'Linear Connection Proofs' correspond to contraction free proofs in sequent systems. On the other hand a relationship with Girard's Linear Logic was suspected.

Pursuing these hypotheses/suspicions, we arrived at the following theorem which characterizes some substructural logics by means of matrices. (Note that we have to employ here matrices which allow multiple occurrences of subrows (and subcolumns) in the same subcolumn (resp. in the same subrow), which coincides with a multiset view of situations in planning.)

Theorem

- A formula F has a *multiplicative derivation*[1] iff the translation of F into a multiset-matrix together with the set of all connections is a complementary matrix.
- A formula F has a *contraction-free* multiplicative derivation iff the translation of F into a multiset-matrix together with a suitable subset S of the set of all connections is a complementary matrix, which in addition is linear (no literal is connected twice) and satisfies a special acyclicity condition.
- A formula F has a *contraction-free* and *weakening-free* multiplicative derivation iff the translation of F into a multiset-matrix together with a suitable subset S of the set of all connections is a complementary matrix, which apart from being linear and acyclic, connects all literals and is minimal.

The third equivalence in the theorem above gives a classification of the multiplicative fragment of pure linear logic. This is one of the answers to various questions concerning the relationship between linear logic and linear connection proofs.

The theorem above can be simplified considerably if we limit ourselves to the language fragment of 'Horn clauses with *conjunctive heads*' which seems to be reasonable for the specification of plan generation problems: In this case the acyclicity condition can be dropped and theorems with contraction-free derivations are characterized by linear complementary matrices.

[1]This means that the multiplicative sequent rules together with exchange, contraction and weakening are used.

A Formal Comparison of Implicit and Explicit Induction

Stefan Gerberding
Technische Hochschule Darmstadt, Fachbereich Informatik
Alexanderstraße 10
64283 Darmstadt
gerberding@inferenzsysteme.informatik.th-darmstadt.de

Abstract

Automated mathematical induction is based on two competing paradigms: Variants of implicit induction by inductive completion resp. proof by consistency (Dershowitz, Fribourg, Huet, Hullot, Kapur, Musser and others) vs. explicit induction (Aubin, Boyer, Bundy, Moore, Walther and others).

After adjusting some formalisms we intend to clarify whether one of the two paradigms leads to a "better" automation of induction than the other.

Criteria have to be developed for comparing proofs in the different methodologies. These criteria have to consider the strength of the computed induction axioms (stronger hypotheses, easier proof obligations) and the (heuristic) mechanisms for selecting one of the several possible inductions.

First we compare explicit induction heuristics (e.g. those of Boyer & Moore and of Aubin) with the implicit method. The (implicit) method of Fribourg is quite similar to explicit induction with regard to the induction heuristic. Fribourgs criteria demand a kind of evaluability for a certain defined function symbol. We found that in general the explicit induction heuristics do not meet these criteria.

We also compare the induction orderings used in the different approaches. While explicit induction makes the induction ordering "explicit", implicit induction rather hides the ordering. For certain term rewriting systems the ordering can be represented by the union of the reduction orderings belonging to the rewriting systems each of which defines one function.

The comparison of the different refutation procedures is much easier obtained as they all rely on the discovery of an *inconsistency witness*. Apparently their main difference is that procedures based on the completion method retain the equivalence induced by the rewriting systems and the equations (up to the last step of the refutation) whereas the methods of explicit induction often use generalization techniques.

A Unifying Framework for Different Function Definition Formalisms Based on Rewriting Techniques

Bernhard Gramlich
Fachbereich Informatik, Universität Kaiserslautern
Erwin-Schroedinger-Str., 67663 Kaiserslautern, Germany
gramlich@informatik.uni-kl.de

Abstract

Recently we have shown some abstract relations between restricted and general termination and confluence properties of term rewriting systems (TRSs for short) (cf. [Gra92]). In particular, we have proved that any innermost terminating overlay system with joinable critical pairs is terminating and confluent. This result provides interesting possibilities of establishing general termination and confluence of TRSs since it can significantly facilitate the task of verifying termination and confluence. Here we show that this result enables us to bridge the gap between different notions of termination and confluence in different frameworks. This is particularly interesting for the theoretical foundation and comparison of various function definition formalisms which are described in literature and used e.g. in inductive theorem provers. It turns out that what is meant by termination of function definitions, for instance in [BM79] and [Wal88], corresponds exactly to termination of innermost rewriting within an equational framework. This relationship and its consequences in theory and practice are discussed. Moreover we sketch the extension and application of our basic results to the analysis of hierarchically structured equational specifications/TRSs (cf. [Gra93]) as well as to the more general case of conditional TRSs.

References

[BM79] R.S. Boyer and J.S. Moore. *A Computational Logic.* Academic Press, 1979.

[Gra92] B. Gramlich. Relating innermost, weak, uniform and modular termination of term rewriting systems. In A. Voronkov, editor, *International Conference on Logic Programming and Automated Reasoning, St. Petersburg*, volume 624 of *Lecture Notes in Artificial Intelligence*, pages 285–296. Springer-Verlag, 1992.

[Gra93] B. Gramlich. Relating innermost, weak, uniform and modular termination of term rewriting systems. SEKI-Report SR-93-09, Fachbereich Informatik, Univ. Kaiserslautern, 1993.

[Wal88] C. Walther. Argument bounded algorithms as a basis for automated termination proofs. In E. Lusk and R. Overbeek, editors, *Proc. of the 9th Int. Conf. on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science*, pages 601–622. Springer-Verlag, 1988.

A Reconstructive Approach towards Proof Presentation

Xiaorong Huang
Fachbereich Informatik, Universität des Saarlandes
Postfach 1150
66041 Saarbrücken
huang@cs.uni-sb.de

Abstract

Most automated theorem provers suffer from the problem that they can produce proofs only in formalisms difficult to understand even for experienced mathematicians. Efforts have been made to transform such machine generated proofs into natural deduction (ND) proofs. Although the single steps are now easy to understand, the entire proof is usually at a low level of abstraction, containing too many tedious steps. Therefore, it is not adequate as input to natural language generation systems.

In this work, we propose a new intermediate representation, called ND style proofs at the *assertion level*. Based on a computational model for *informal* mathematical reasoning, assertion level deduction steps are designed to capture more semantic derivations, which intuitively speaking, follow by applying a previous result or definition. We further show that the assertion level steps can be justified by domain-specific inference rules, and that these rules can be represented compactly in a tree structure. Using assertion level constructs as the goal language, a procedure is devised which substantially shortens ND proofs by abstracting them to the assertion level. Technically, such abstraction involves a *reconstruction* of the proof using assertion level constructs.

The above mentioned algorithm is implemented in a proof presentation system called *PROVERB*, and tested within the proof development environment Ω-MKRP. Despite its simpleness, the current algorithm substantially shortens input ND proofs of a broad class. Most significant reduction is observed with input proofs which are essentially direct proofs, but containing machine generated detours and redundancies. As a typical example, a machine generated ND proof with 134 lines is abstracted to a proof of 15 lines. The algorithm also works well on neatly structured ND proofs. In these cases, the reduction factor depends on the average depth of the terms in the definitions and theorems involved in the proof. Since mathematicians usually avoid using both too trivial and too complicated definitions and theorems, a quite stable reduction factor (about two thirds in length) is normally achieved. Finally, the algorithm performs very poorly on machine generated proofs which are mainly indirect, i.e., in most of the lines only bottom is derived. Despite of a reduction factor of about one third in length, the remaining proof lines are still largely at the level of calculus rules and the proof is therefore still too tedious.

Entscheidungsverfahren für modale Logiken

Jörg Hudelmaier
WSI, Universität Tübingen
Sand 13
Tübingen
joerg@logik.informatik.uni-tuebingen.de

Abstract

We present improved decision procedures for the well known PSPACE-complete modal logics K, T and S4. The decision procedures are based on so called contraction free sequent calculi, i.e. calculi for which all possible deductions of a given sequent are bounded in depth by a recursive function in the length of that sequent. The growth rate of the recursive function then yields an upper bound for the space complexity of the logic considered. Using reduction of the sequents to modal sequents we obtain linearly growing recursive functions for the logics K and T and a quadratically growing function for S4. For K and T the calculi we use for obtaining these bounds are straightforward adaptations of well known (e.g. Tableaux-) calculi. But for S4 adapting the usual calculus would not result in a contraction free sequent calculus. We therefore have to consider a completely different calculus which comes in two versions, the first one having rules with arbitrarily many premisses and dealing with clausal sequents only, the second one being derived from it, but of more traditional shape, i.e. with at most two premiss rules and using the full language.

Solving Simplification Ordering Constraints

Patricia Johann and Rolf Socher
FB Informatik, Universität Saarbrücken and MPI Informatik
Im Stadtwald, 66123 Saarbrücken
pjohann@cs.uni-sb.de, socher@mpi-sb.mpg.de

Abstract

The concept of well-founded orderings plays an important role in Automated Deduction and Term Rewriting. It is fundamental for the development of termination proofs or theorem proving strategies that serve to restrict the search space in saturation processes such as Knuth-Bendix Completion or Ordered Resolution.

During the last years, there has been an increasing interest in the use of constraint techniques in Automated Deduction. They proved useful as a means to separate the indeterministic proof search from deterministic computations like unification, order sorted unification, or computations in models such as the reals. Most recently, also ordering constraints have been used in deduction systems (Nieuwenhuis & Rubio, CADE 92).

The basic constraint problem is: Given a set

$$\{(s_1, t_1), ..., (s_n, t_n)\}$$

of termpairs over a signature sig, is there a substitution σ and an ordering $\succ$ such that $s_i\sigma \succ t_i\sigma$ holds for $i = 1, \ldots, n$? Depending on the underlying signature sig, two cases can be distinguished: the *fixed signature case* where σ is a sig-substitution, and the *extended signature case* where σ ranges over an arbitrary extension of sig. For instance, given the signature consisting of a single constant a, the constraint (x, a) is unsolvable in the fixed signature case. In the extended signature case, however, a new constant b may be introduced such that the substitution $\{x \mapsto b\}$ together with an ordering $\succ$ that satisfies $a \succ b$ solves the problem.

Comon (LICS 1990) showed that the problem is decidable if $\succ$ is interpreted as a *lexicographic path ordering* (LPO). Unfortunately, his method is of double exponential time complexity. Other, less complex methods to solve LPO-constraints have been proposed, but it seems that the problem is inherently NP-complete. The major advantage of using *simplification ordering constraints* rather than *LPO constraints* is the polynomial complexity of the constraint solving algorithm. In this talk, we provide a simple decision procedure that has polynomial time complexity in the extended signature case. Moreover, we show that the problem is NP-complete if $\succ$ is restricted to *total* simplification orderings. Finally, we give a constraint solving procedure for the fixed signature case.

Higher-Order Resolution with Combinators

Michael Kohlhase
Fachbereich Informatik, Universität des Saarlandes
Im Stadtwald
66041 Saarbrücken
`kohlhase@cs.uni-sb.de`

Abstract

In this talk we present a machine-oriented calculus for higher-order logic that is based on Curry's combinatory logic. This logic System $\mathcal{CL}$ is a formulation of higher-order logic that is equivalent to Church's simply typed λ-calculus $\mathcal{LC}$, which in contrast to $\mathcal{LC}$ does not need bound variables. Unfortunately this formulation is very difficult to read; but there are well-known transformations between the systems. Thus a user need only manipulate $\mathcal{LC}$ formulae even when using $\mathcal{CL}$ as a working language for a deductive system.

In recent years Dougherty and Johann have developed a series of unification algorithms for $\mathcal{CL}$, which we will use for our calculus. Instead of – as these authors suggest – using these algorithms as part of a $\mathcal{LC}$ prover which necessitates transforming the relevant formulae before and after each unification step, we present a resolution calculus totally inside $\mathcal{CL}$ so that only the input formulae and the completed proof for a deductive system have to be transformed.

We expect that a higher-order deduction system for $\mathcal{CL}$ can be realized with the technology developed for first-order deduction systems and therefore has great advantages from the implementation point of view. However the practical efficiency of a $\mathcal{CL}$ deduction system has still to be investigated by experimentation.

There is reasonable hope, that the unification algorithms mentioned above will be more suitable for integrating techniques from unification theory for first-order languages than the algorithms for $\mathcal{LC}$. This fact possibly constitutes a great potential for development of deduction systems for higher-order logic.

Optimizing Proof Search by Machine Learning Techniques

Thomas Kolbe, Christoph Walther
Technische Hochschule Darmstadt, Fachbereich Informatik
Alexanderstr. 10
64283 Darmstadt
{kolbe,walther}@inferenzsysteme.informatik.th-darmstadt.de

Abstract

This research aims to develop a *learning component* for an induction theorem prover. After a system has computed an induction axiom for a given conjecture, the induction formulas, i.e. the base and step cases, have to be proved. This requires some search control for selecting useful formulas among the given axioms and lemmata, and also for deciding where and in which proof step a selected formula and/or an induction hypothesis is applied. There are several proposals to solve this control problem, e.g. support by a human user by means of "hints" and "rewrite lemmata", the "rippling technique" (developed in Alan Bundy's research group) or Dieter Hutter's "C-term method" which both look for similarities of certain patterns in the conjecture and the available axioms and lemmata to guide the proof.

Our idea for tackling this problem is to analyse a proof of a conjecture and then to compute the relevant features of this proof in terms of "applied" axioms and lemmata. This means that "abstractions" of the leafs of a proof tree are computed, where the key of success lies in the generality of these "abstractions". The approach follows closely the paradigm of *explanation based learning*, where a learning component computes solution schemas by analysing the outcomes of some problem solver. The problem solver is an induction theorem prover in our application, which may be supported in part or completely by a human. If a new conjecture has to be proved, it is tested firstly whether the new conjecture is "similar" to a previously proved conjecture, and in this case it is attempted to verify the instances of the features learned from the previous proof instead of verifying the original conjecture.

The usefulness of our proposal depends on the frequency of proof-reuses in realistic applications, and therefore the question for usefulness can be answered only after experiments of appropriate size have been carried out with an implementation (which is in preparation). To date we have studied only theorem proving problems which are quite easily solved by completion based systems. However, results obtained by manual experiments seem to be promising. These results are based on our proof analysis technique, on our notion of abstracting formulas and on our representation of previously computed proofs.

Generation and Modification of Transformation Orderings

Jochen Nessel
Fachbereich Informatik
Postfach 3049
67653 Kaiserslautern
e-mail: nessel@informatik.uni-kl.de

Abstract

The subject of this abstract is the proof of termination of Term Rewriting Sytems (TRS) R. This can be done using two other TRS, S and T, such that S$\cup$T terminates by a Recursive Path Ordering [2] (RPO). S together with T and $>_{RPO}$ defines an instance of a Transformation Ordering (TO), if certain conditions are fulfilled. The proof for R is done by checking each rule $l \rightarrow r$ of R in the following way. Both sides of the rule are reduced to their respective normal forms using T. These normal forms are then compared using either $\rightarrow_S$ or $>_{RPO}$.

So the problem is to find suitable S and T for a given R. There exist implementations to find S, whenever T and R are given, but usually it is more difficult to find the 'transformation' T. We developed and implemented a heuristic based algorithm $\mathcal{A}$ to partially solve this problem. Its only input is the TRS R. If it halts with output S, T, $>_{RPO}$, then S$\cup$T terminates by $>_{RPO}$, S and T have the required properties and R is terminating. The proceeding is very intuitive: $\mathcal{A}$ first checks every $l \rightarrow r \in R$ with $>_{RPO}$. If this succeeds, then $l \rightarrow r$ is put into S. If the test fails, r will be analysed and the subterms preventing an orientation will be identified. Then for each such subterm a transforming rule will be created. The orientation of the transformed rule is achieved by introducing new (minimal) function symbols. The transformations are then added to T. When all rules in R are oriented using this technique, the resulting TRS S and T will be modified to fulfill the required restrictions. This process need not terminate, but if does, S and T prove the termination of R. This simple approach was already successful for some quite hard-to-prove TRS. Nevertheless it failed on other TRS which are orientable with TO. So we developed several additional heuristics to improve the power of the algorithm. Some of these new heuristics and a known one were implemented.

Another question raises when one investigates TO: Is it possible to weaken the conditions, i.e. has T to be variable preserving and left linear? It is possible to drop left linearity when the definition of TO is slightly changed, but S now has to be right linear and confluent, among other restrictions. With the left linearity droped, it is for example possible to prove the rule f(0,1,x)$\rightarrow$f(x,x,x) terminating with T = {f(x,x,x) $\rightarrow$ q(x) } and S = {f(0,1,x) $\rightarrow$ q(x)}.

[2] References are given in Jochen Nessel, Implementierung und Erweiterung eines Verfahrens zur Erzeugung von Transformationsordnungen, Diplomarbeit, Univ. Kaiserslautern, 1993.

Decidable Higher-Order Unification and Second-Order Narrowing

Christian Prehofer
Institut für Informatik
Technische Universität München
80290 München
E-mail: prehofer@informatik.tu-muenchen.de

Abstract

Second-order unification is undecidable in general. Dale Miller showed that unification of so-called higher-order patterns is decidable and unitary. A simply typed λ-term s is a higher-order pattern, if all its free variables only have distinct bound variables as arguments. For instance, $\lambda x, y.F(x, y), \lambda x.f(G(\lambda z.x(z)))$ are patterns, but $\lambda x, y.F(a, y), \lambda x.G(H(x))$ are not. We show that the unification of a linear higher-order pattern s with an arbitrary second-order term that shares no variables with s is decidable and finitary. The first step of the proof is a termination ordering for the known higher-order pre-unification algorithms with some minor modifications. If this algorithm succeeds, there can remain equations with variables as outermost symbols on both sides (flex-flex pairs), e.g. $\lambda x, y.F(x, y) = \lambda x, y.G(H(x))$. In the general case, there can be infinitely many incomparable solutions to flex-flex pairs, but there always exists a solution. In our case these pairs fall into a certain class and can be finitely solved.

We show that some extensions of this unification problem are still decidable, but may not finitary. For instance, if the arguments to free variables are ground second-order terms and not only bound variables, the pre-unification problem is still solvable and finitary. Unifying two second-order terms, where one term is linear, is undecidable if the terms contain bound variables and decidable if they don't.

An interesting application of these results is to attempt second-order narrowing with decidable unification. This could be the basis for second-order functional logic programming languages. For instance, we can easily express differentiation by rules like $(sin(F(x)))dx \to cos(F(x)) * (F(x)dx)$, where F is a variable of functional type. We will show that narrowing cannot be extended in the straightforward manner to the second-order λ-calculus; we have to cope with the conversions of λ-calculus and with variables of higher type.

Other areas of applications are theorem provers which work with higher-order unification and possibly second-order unification problems arising in type inference.

Lazy Generation of Induction Hypotheses

Martin Protzen
Technische Hochschule Darmstadt, Fachbereich Informatik
Alexanderstr. 10
64283 Darmstadt
protzen@inferenzsysteme.informatik.th-darmstadt.de

Abstract

The degree of mechanization of an inductive theorem prover depends on the prover's ability to generate relevant induction hypotheses. Automated induction proofs following the explicit induction paradigm (as opposed to the implicit induction paradigm which evolved from the Knuth-Bendix Completion Procedure) have since the fundamental work of Boyer & Moore [Boyer&Moore 79] constructed inductions schemes by

- collecting induction schemes suggested by recursive functions,
- manipulating induction schemes (heuristically),
- combining different induction schemes, and finally
- selecting one of the surviving schemes.

Although successful this process involves many difficulties and can lead to either waste of resources or decisions which prevent that a proof for (inductively) true formulas can be found. Recent research [Walther 92, Stevens 89] has proposed enhancements but sometimes the conventional techniques still fail to provide relevant induction hypotheses.

The proposed method delays the computation of hypotheses until they are applicable — and this is the reason to call the method "lazy". First, an induction heuristic selects appropriate subterms of the proposition to prove and evaluates these symbolically. Subsequently the evaluated proposition is manipulated by techniques similar to the rippling method [Bundy 92] (or the context approach [Hutter 90]) until the manipulated proposition matches a generalized template for induction hypotheses. The rippling method is extended by using induction variables as additional sinks, although only to a limited extent. Thus, not only induction schemes which are suggested in the conventional approach can be generated, but also additional ones. The well-foundedness of the computed schemes can be proved reusing termination proofs for the functions involved in the proposition. Finally, the method can be extended to include goal-directed generalizations of the proposition when the rippling process becomes blocked.

The approach successfully generates induction hypotheses even for hard problems, e.g. the Gilbreath Card Trick [Gardner 60].

Das Beweissystem KoMeT

Thomas Rath, Stefan Brüning, Wolfgang Bibel, Uwe Egly

FG Intellektik,
FB Informatik, Technische Hochschule Darmstadt
Alexanderstraße 10, D–64283 Darmstadt (Germany)
E-mail: thomas,stebr,bibel,uwe@intellektik.informatik.th-darmstadt.de

Abstract

KoMeT ist ein vollständiges und konsistentes Beweissystem für die Prädikatenlogik erster Stufe. Das bei der Konzeption und Entwicklung von KoMeT verfolgte Ziel, ist es ein berechnungsadäquates Beweissystem zu erhalten, das grob gesagt einfache Theoreme schneller beweist als schwierige. Ein Beweissystem, das dieses Kriterium erfüllen soll, muß verschiedenste Mechanismen zur Behandlung unterschiedlicher Problemklassen enthalten. Die konzeptionell erstrebenswerte Einheitlichkeit wird durch eine Einbettung dieser verschiedenen Mechanismen in die Konnektionsmethode erreicht. Hierzu gehören die Ausnutzung von Reduktionen, Indizierungstechniken, Verfahren zur Suchraumbeschränkung, verschiedene Suchstrategien, der Einsatz von Theoriekonnektionen, die Verwendung von Induktion, eine effiziente Lemmabehandlung, die Berücksichtigung von Beweisplänen sowie die Generierung von Beispielen und Gegenbeispielen. Da das Zusammenspiel all dieser verschiedenen Mechanismen nicht nur theoretisch, sondern auch experimentell erforscht werden muß, ist es von besonderer Bedeutung, möglichst leicht Modifikationen am System KoMeT durchführen zu können. Hieraus ergibt sich auch, daß KoMeT betont prototypischer Natur ist. Die Techniken, die sich in KoMeT als besonders vorteilhaft erweisen, können anschließend in einer programmiertechnisch optimierten Form in bestehende Systeme wie z.B. SETHEO integriert werden.

Schwerpunktmäßig wurden bisher Reduktionen, Indizierungstechniken, Verfahren zur Suchraumbeschränkung, verschiedene Suchstrategien, der Einsatz von Theoriekonnektionen (Gleichheit) und die Verwendung von Induktion untersucht. Im Augenblick enthält KoMeT neben einem Modul zur Normalformtransformation einen umfangreichen Satz an Reduktionen. Weiterhin ist es mit Hilfe von Datenbanktechniken möglich, ähnliche Teile einer Formel zusammenzufassen und im anschließenden Beweis gemeinsam zu bearbeiten. Bekannte Verfahren zur Suchraumbeschränkung, verschiedene Suchstrategien, sowie eine auf Paramodulation beruhende Variante der Gleichheitsbehandlung sind ebenfalls verfügbar. Desweiteren ist es möglich, Beweise mit Hilfe von struktureller Induktion über dem Standardmodell der natürlichen Zahlen zu führen. In der nächsten Zeit ist vor allem daran gedacht, eine effiziente Behandlung von Lemmata, weitere Suchstrategien und Theorien, sowie eine komfortable Benutzeroberfläche in KoMeT zu integrieren.

On Gröbner Bases in Monoid and Group Rings

Klaus Madlener
Birgit Reinert
Fachbereich Informatik, Universität Kaiserslautern
W–67653 Kaiserslautern, Germany
email: madlener@informatik.uni-kl.de, reinert@informatik.uni-kl.de

Abstract

Following Buchberger's approach to computing a Gröbner basis of a polynomial ideal in polynomial rings, a completion procedure for finitely generated right ideals in $\mathbf{Z}[\mathcal{H}]$ is given, where $\mathcal{H}$ is an ordered monoid presented by a finite, convergent semi–Thue system (Σ, T). Taking a finite set $F \subseteq \mathbf{Z}[\mathcal{H}]$ we get a (possibly infinite) basis of the right ideal generated by F, such that using this basis we have unique normal forms for all $p \in \mathbf{Z}[\mathcal{H}]$ (especially the normal form is 0 in case p is an element of the right ideal generated by F). As the ordering and multiplication on $\mathcal{H}$ need not be compatible, reduction has to be defined carefully in order to make it Noetherian. Further we no longer have $p \cdot x \rightarrow_p 0$ for $p \in \mathbf{Z}[\mathcal{H}], x \in \mathcal{H}$. Similar to Buchberger's s–polynomials, confluence criteria are developed and a completion procedure is given. In case $T = \emptyset$ or (Σ, T) is a convergent, 2–monadic presentation of a group providing inverses of length 1 for the generators or (Σ, T) is a convergent presentation of a commutative monoid, termination can be shown. So in this cases finitely generated right ideals admit finite Gröbner bases. The connection to the subgroup problem is discussed.

References

[MaRe93a] K. Madlener and B. Reinert. *On Gröbner Bases in Monoid and Group Rings.* SEKI Report SR-93-08. Universität Kaiserslautern.

[MaRe93b] K. Madlener and B. Reinert. *Computing Gröbner Bases in Monoid and Group Rings.* Proc. ISSAC'93. pp 254-263.

On the Problem of Generating Small Convergent Systems

Klaus Madlener , Andrea Sattler-Klein
Fachbereich Informatik, Universität Kaiserslautern
D-67653 Kaiserslautern, Germany
Friedrich Otto
Fachbereich Mathematik/Informatik, Gesamthochschule Kassel
D-34109 Kassel, Germany

Abstract

Given a finite string-rewriting system R on some alphabet Σ and an admissible total well-founded ordering $>$ on Σ^* as input, the Knuth-Bendix completion procedure tries to transform the system R into an equivalent finite system S such that S is convergent and compatible with the given ordering. Since it is undecidable in general whether or not such a system S exists termination of the completion procedure is undecidable as well. This means that there is no recursive function that can serve as an apriori upper bound for the running time of the completion procedure for those inputs $(R, >)$ that lead to termination. Hence, for each recursive function $f : \mathbb{N} \to \mathbb{N}$, there exists a sequence of finite string-rewriting systems $(T_n)_{n \in \mathbb{N}}$ such that, given the system T_n and the ordering $>$ as input, the completion procedure will terminate eventually, but it will perform more than $f(size(T_n))$ steps until then. One obvious reason for this phenomenon to occur is the fact that the resulting finite convergent system S_n that is generated from T_n can be extremely large. We will show, however, that even in the case that the resulting finite convergent system S_n is small, i.e., it is approximately of the size of the input system T_n, the completion procedure may perform extremely many steps. More precisely, we will construct a sequence $(R_{n,m})_{n,m \in \mathbb{N}}$ of normalized string-rewriting systems on a fixed finite alphabet Σ such that, for all $n, m \in \mathbb{N}$,

1. $R_{n,m}$ contains 44 rules, it is of size $O(n+m)$, and it is compatible with a length-lexicographical ordering $>$ on Σ^*, but
2. given the system $R_{n,m}$ and the ordering $>$ as input, the Knuth-Bendix completion procedure will generate more than $A(n, m)$ intermediate rules before a finite convergent system $S_{n,m}$ of size $O(n + m)$ is obtained. Here A denotes Ackermann's function.

References

[1] K. Madlener, F. Otto, A. Sattler-Klein. Generating small convergent systems can be extremely hard. *Proc. 3rd International Symposium on Algorithms and Computation*, Nagoya, Japan, December 1992, LNCS 650, 299-308.

[2] K. Madlener, A. Sattler-Klein, F. Otto. On the problem of generating small convergent systems. *Journal Symbolic Computation*, (to appear).

Groundreducibilitytests —
Even for Nonlinear Term Rewriting Systems

Klaus Schmid
FB Informatik
Universität Kaiserslautern
67653 Kaiserslautern

Abstract

A new method for generating ground reduction test sets that works even for nonlinear and many-sorted rewriting systems is proposed. It combines the efficient process of generation as suggested by Bündgen with small test sets as generated by the method of Kapur, Narendran and Zhang.

This method has been extended to cover nonlinear term rewriting systems, too. The influence that nonlinear variables have on the form of the test set is described using so-called "constraints". They are an adaption of inequalities — as they have been proposed by Comon — to this particular problem. For resolving these constraints the notion of transnormality as proposed by Kounalis has been significantly reworked and refined. Therefore a completely new theory had to be created showing the correctness of these ideas. Moreover, this theory covers the described method for linear term rewriting systems guaranteeing a seamless transition between the treatment of linear and nonlinear term rewriting systems.

Other advantages include simple treatment of many-sorted rewriting systems and an explicit discrimination between criteria necessary during the generation process and conditions necessary for the applicability of the resulting test sets.

We expect that an implementation of this method would be very efficient because no usage of test sets in the process of generation is required.

Analysis and transformation of equational proofs in a distributed environment

Stephan Schulz
FB Informatik
Universität Kaiserslautern
67653 Kaiserslautern

Abstract

With the advent of more powerful computers and better algorithms automatic proof systems have been used to prove more and more difficult problems. However, while the power of these proof systems has risen steadily, the general acceptance of proofs generated by them still leaves a lot to be desired. In this talk we try to introduce a way to present completion based equational proofs in a way that can be read and understood by most people with a minimal mathematical background.

To transform the proof into this easy to read format we first obtain a complete step by step listing of the proof process in a language called PCL (proof communication language). While this is a simple task for a sequential proof system it is in no way trivial for a distributed theorem prover, because the necessarily extensive output interferes with timing considerations. However, it is well worth the effort. By analysing the proof listings, valuable information regarding the proof process and the heuristics of the prover can be gained.

PCL listings can be structured to reveal important proof steps by introducing lemmata. We develop several different criteria to recognize potential lemmata. The structured PCL listings can then be transformed into a calculus using only equational chains. Proofs in this calculus can be easily adapted to conform with human reading conventions.

On the Automatic Generation of Polynomial Orderings for Proving the Termination of Term Rewriting Systems

Joachim Steinbach
SFB 314 (D4), Universität Kaiserslautern
Postfach 3049
67653 Kaiserslautern
steinba@informatik.uni-kl.de

Abstract

As programs *term rewriting systems* (TRSs) have a very simple syntax and their semantic is based on equalities that are used as reduction rules with no explicit control. For this purpose it is essential that a TRS has the property of *termination* . Most of the various methods for proving termination of TRSs are based on *reduction orderings* which are well-founded, compatible with the structure of terms and stable wrt substitutions. One way of constructing reduction orderings consists of the specification of a well-founded set $(\mathcal{W}, \succ)$ and a mapping φ from the set of terms into $\mathcal{W}$, such that $\varphi(s) \succ \varphi(t)$ whenever t can be derived from s. *Polynomial orderings* are based on the set of polynomials over $\mathbb{N}$ $(= \mathcal{W})$ where φ denotes a polynomial interpretation and $\succ$ represents an ordering on polynomials.

One of the main problems concerning polynomial orderings is the *choice of the right interpretation* for a given TRS. The object of this paper is to present new insights into the automatic generation of termination proofs using polynomial orderings. We have developed an algorithm based on the *method of complete description* (a linear programming technique) for finding a polynomial interpretation of a given TRS provided that this system can be oriented using polynomials of a special form. This technique is restricted to so-called *simple-mixed* polynomials because it is very difficult to compare two *general* polynomials. However, since 96% of the interpretations used for the orientation of the 320 tested examples (more than 1700 reduction rules) are simple-mixed it is an acceptable restriction. According to the algorithm of U.Martin for generating an appropriate weight function for the Knuth-Bendix ordering, we transform the set of rules into a set of *linear* inequalities based on the coefficients of the interpretations wrt common variables. Then there exists a relatively simple algorithm, the so-called *method of complete description* for deciding whether a system of *linear* inequalities has a solution. Unfortunately, we do not have *linear* inequalities, initially. In order to apply the method of complete description we will transform more general inequalities to linear ones by 1. approximating each side to exactly one product and then 2. applying a logarithmic function to the resulting products.

We have implemented the algorithm discussed above. The implementation does not require any user interactions. Note that the presented technique is not a decision procedure. However, it is very useful in practice as confirmed by the following statistics: We have applied the algorithm to 242 TRSs (which are all orientable with the help of simple-mixed polynomials). For 228 TRSs (94.2%) the method was successful.

Minimal Resolution

Christoph Weidenbach
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
weidenb@mpi-sb.mpg.de

Abstract

The idea of minimal resolution is to restrict the resolution rule and the factorization rule to operate only on “minimal” literals. This idea is not new and is e.g. implemented by lock resolution or ordered resolution. For lock resolution the minimal literal of a clause is the one with the lowest index. For ordered resolution the minimal literals of a clause are the smallest literals wrt. a term ordering.

I assume an abstract, total function which maps a clause to its set of minimal literals. Application of resolution is restricted to the minimal literals of the parent clauses. The factorization rule is applicable to a minimal literal of the parent clause and a different one. If the minimality function satisfies certain conditions wrt. instantiation of clauses, splitting of clauses, resolution, and factorization minimal resolution is complete. The completeness of lock resolution and ordered resolution are instances of this result. In addition, I present some new minimality criteria and show their completeness.

Constructor-Based Inductive Validity in Positive/Negative-Conditional Equational Specifications

Claus-Peter Wirth, Bernhard Gramlich, Ulrich Kühler, Horst Prote.
Fachbereich Informatik, Universität Kaiserslautern,
D-67663 Kaiserslautern, Germany.
wirth@informatik.uni-kl.de

Abstract

We present results from our SEKI-Report SR–93–05 in two talks:

1. Positive/Negative-Conditional Equational Specifications
(Ulrich Kühler)
2. Notions of Inductive Validity
(Claus-Peter Wirth)

Both talks deal with algebraic specifications given by finite sets R of positive/negative-conditional equations (i. e. universally quantified first-order implications with a single equation in the succedent and a conjunction of positive and negative (i. e. negated) equations in the antecedent). The class of models of such a specification R does not contain in general a minimum model in the sense that it can be mapped to any other model by some homomorphism. We present a constructor-based approach for assigning appropriate semantics to such specifications. We introduce two syntactic restrictions: firstly, for a condition to be fulfilled we require the evaluation values of the terms of the negative equations to be in the constructor sub-universe which contains the set of evaluation values of all constructor ground terms; secondly, we restrict the constructor equations to have "Horn"-form and to be "constructor-preserving". A reduction relation for R is defined, which allows to generalize the fundamental results for positive-conditional rewrite systems, which does not need to be noetherian or restricted to ground terms, and which is monotonic w. r. t. consistent extension of the specification. Under the assumption of confluence, the factor algebra of the term algebra modulo the congruence of the reduction relation is a minimal model which is (beyond that) the minimum of all models that do not identify more objects of the constructor sub-universe than necessary and which establishes one of the four notions of inductive validity of Gentzen clauses we discuss.